Logic Analyzer

Intronix Logicport 34 Channel Logic Analyzer

LogicPort Setup

- You will need
  - Intronix LogicPort logic analyzer (from now on referred to just as the logic analyzer)
  - LogicPort software installed on your computer
    - http://www.pctestinstruments.com/downloads.htm
  - Something to test (microcontroller, simple logic chip)

Information from Intronix

- Main Website
  - http://www.pctestinstruments.com/

- **Help in the LogicPort software**
  - Super helpful (most of the time)
  - Tutorials and specifics of all the menus options, etc.
  - Help->Contents… (or just hit F1 or the question mark icon)

- Example setups (signals, groups, pretty colors)
  - File->Open->C:/Program Files/LogicPort/Projects/Examples/…

LogicPort Basics

Logic Analyzer Intro

- 38 total wires
  - 34 data channels (rainbow wires)
  - 2 clock channels (white wires)
  - 4 ground channels (grey wires)
- Up to 500MHz sampling rate
- Input impedance: 200K ohms
- The box holds an FPGA which does all of the sampling

Logic Analyzer Intro

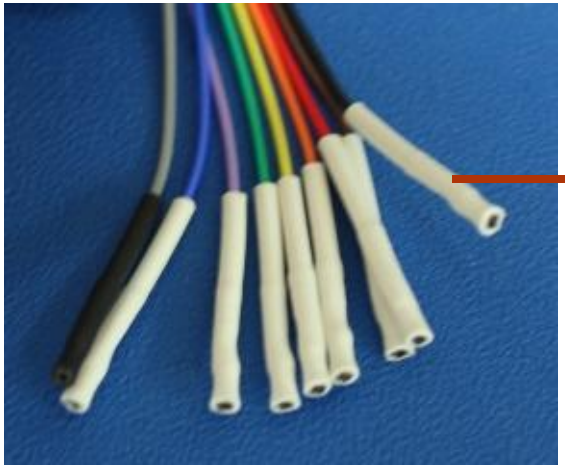
- A logic analyzer only works on digital signals (HIGH or LOW)
- It is good for debugging logic circuits (including microcontrollers, logic chips, ADC converters, etc.)
- Most logic analyzers work in the following manner:
  - Set up the voltages for high and low states
  - Set up a the trigger situation (e.g. a signal going from high to low, a parallel bus being equal to 0x40, etc.)
  - When the trigger situation occurs, display the buffer
  - In a logic analyzer samples are being taken at certain rate which fill up a buffer. This buffer is then displayed for the user to look at.

Ground Channels

- 4 ground channels (grey wire with black tip)
- You should **always** use a ground channel
  - **General rule: use one ground wire per 8 data wires**
- Not grounding the logic analyzer will usually give you crappy results
  - The reference voltage for the sampling is in effect floating

Data Channels

- 32 data channels
- The grey and white wires are NOT data wires.
- The wires colors and tip colors match with the wire ID colors displayed in the software
  - Black wire with a white tip is D0 (the black letters with white background)



Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Clock1	CLK1	L	X		0
Clock2	CLK2	L	X		0
- Data[31..0]				•	0h
Data0	D0	L	X	↑	0
Data1	D1	L	X		0
Data2	D2	L	X		0
Data3	D3	L	X		0
Data4	D4	L	X		0
Data5	D5	L	X		0
Data6	D6	L	X		0
Data7	D7	L	X		0
Data8	D8	L	X		0

Clock Channels

- 2 Clock Channels (white wires, CLK1, CLK2)

- Used for two main reasons:
  - Determine a clock frequency using the logic analyzer
  - OR
  - Use the external clock source to decide when to sample (see the sampling section for more information)
- You don’t always need a clock source, the logic analyzer has an internal clock source

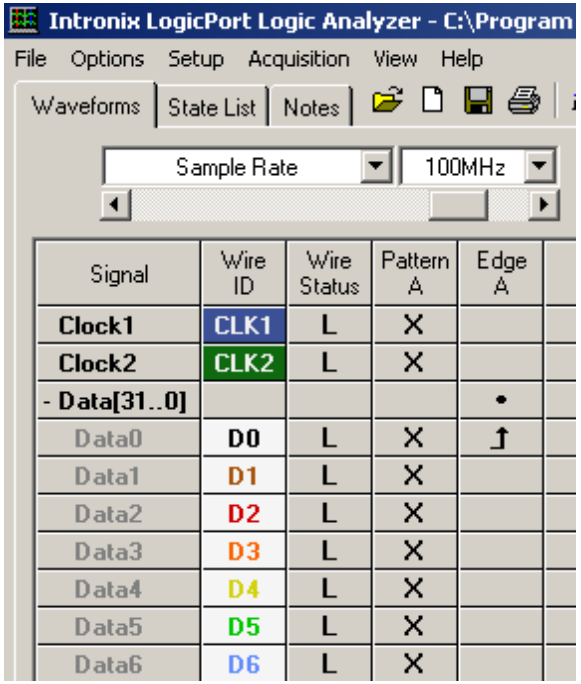
Software Overview

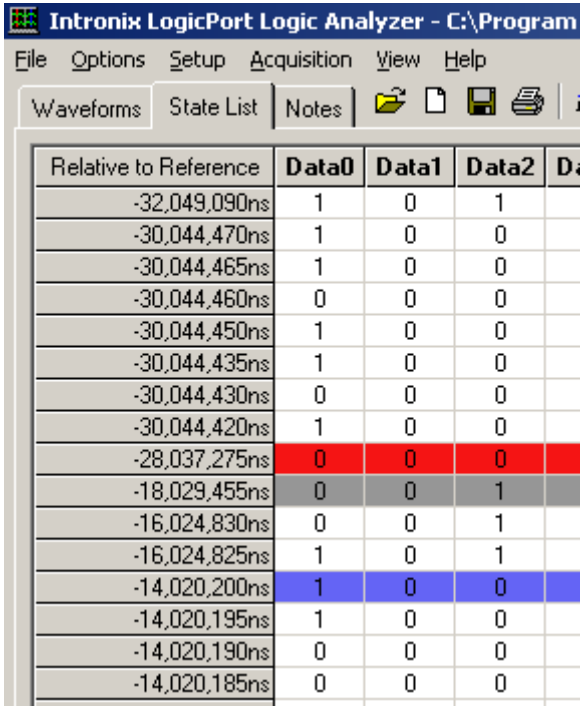
Software Tool Bar

- Sample Mode Setup
- Trigger Setup
- Measurement Setup
- Acquisition Modes
  - Single acquisition (wait for trigger, then take one acquisition
  - Continuous acquisition (wait for trigger, then keep taking samples
  - Stop (to get out of continuous acquisition)
  - Trigger immediate (take a full sample/buffer) now, regardless of trigger situation
- Quick Travel
- Zoom In/Out, Fit

Software: Views

- Two main views: waveforms and state list

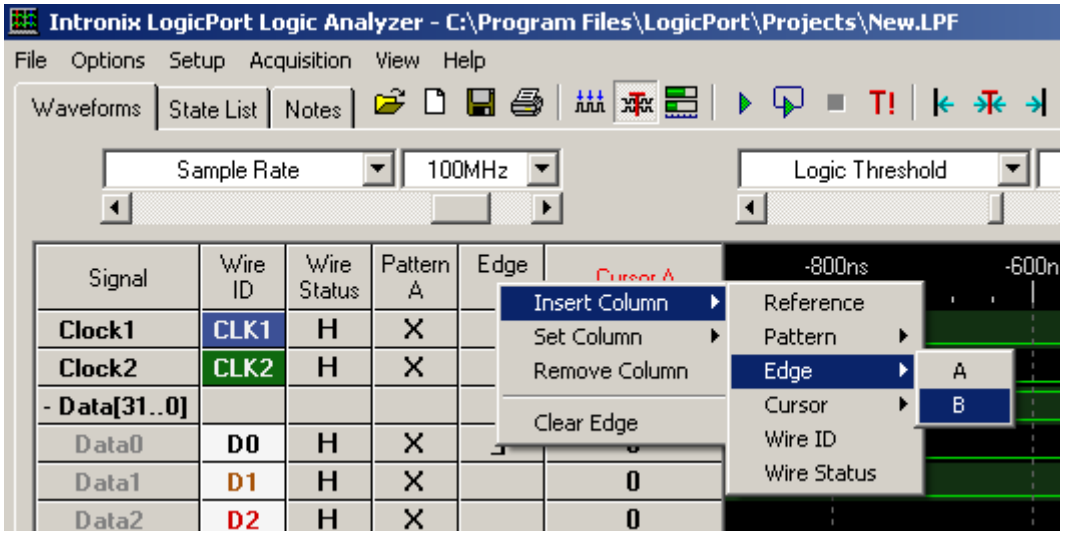




- Same data, one is in graphical format, one is a table format

Waveform View: Columns

- You can add/remove columns by right clicking on the column names
  - Useful for more advanced triggering, looking at cursor values, etc.



Setting Up an Acquisition

Software Setup

- Creating a ‘new file’
- Adding/taking away signals
- Adding/taking away groups
- Renaming signals/groups
- Group settings
  - Format
  - Order (MSB,LSB)
  - Analog/Digital
  - Color

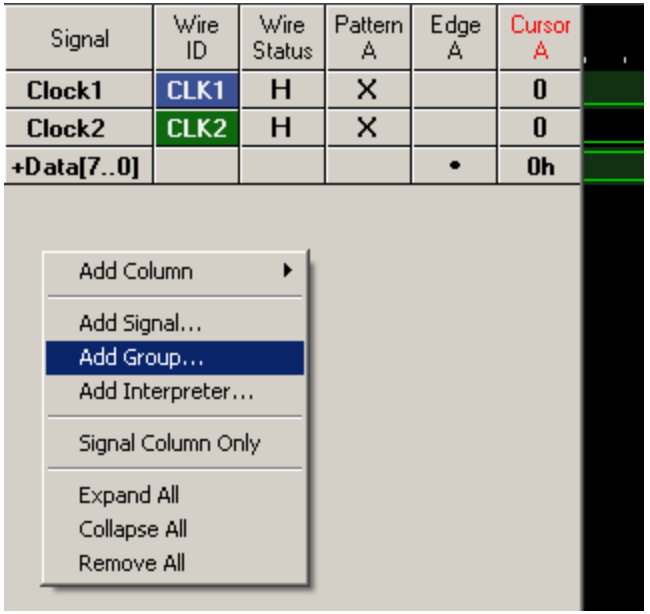
New Setup

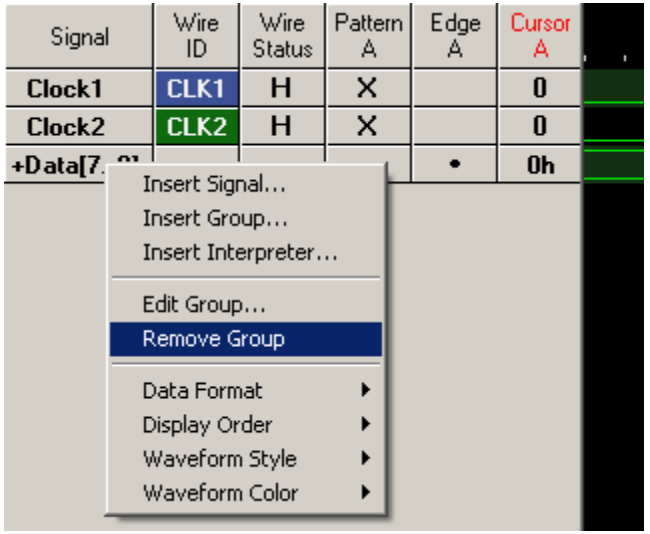
- You will want to create a new setup, which is called a ‘new file’
  - This file, if you save it, will keep all of your settings (logic threshold, number and choices of signals/groups, interpreters and interpreter settings, colors, etc.)
  - You can then reload the file when you want to get the same setup back.
- Usually, you will want to remove the standard group (Data[31..0]) and the clock signals and then add back only what you want.

Signals and Groups

- A signal (a single wire) and a group (a set of wires) can be **added** or taken away by right clicking in the open area below the signals

- A group or signal can be taken away by right clicking on the signal or group





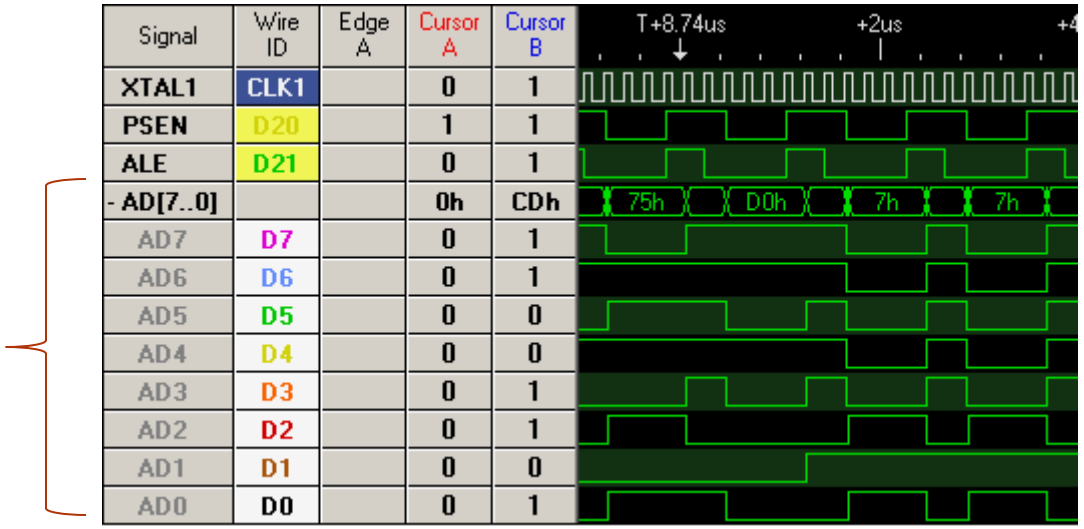
Signals and Groups

- Default groups are 8-bit, 16-bit and 32-bit groups, starting from the Data0 signal and going up
  - You can create your own group (if you for example needed a 12-bit bus)
    - Add Group->Create button
- You can have multiple groups (which use the same signals), but not the same group twice
- You can use a group that has signal Data0, and then also have Data0 be its own signal

Groups

- Groups are useful when looking at parallel data buses because the software will interpret the data for you (you don't have to do the bit-to-hex conversion)

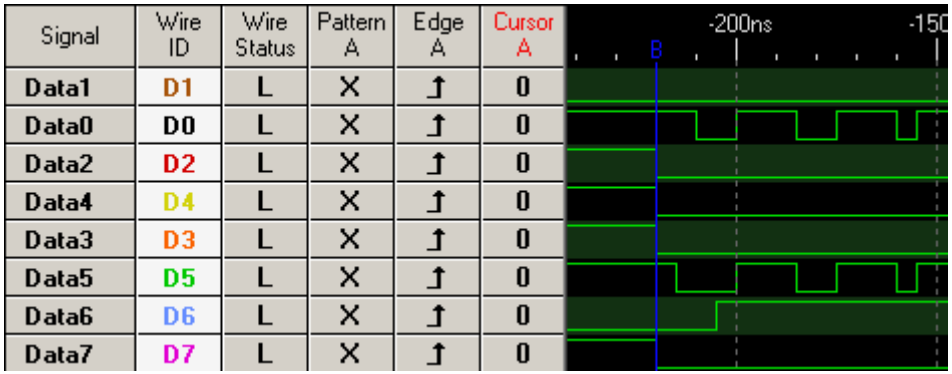
8-bit parallel bus, which is looked at using the Data[0..7] group. You can collapse the group by clicking the little minus sign.



Reordering and Renaming

- A couple of things will make your setup more useful
  - Put signals/groups in a reasonable order
    - Don’t put stuff in strange order: data[5], data[0], data[6]

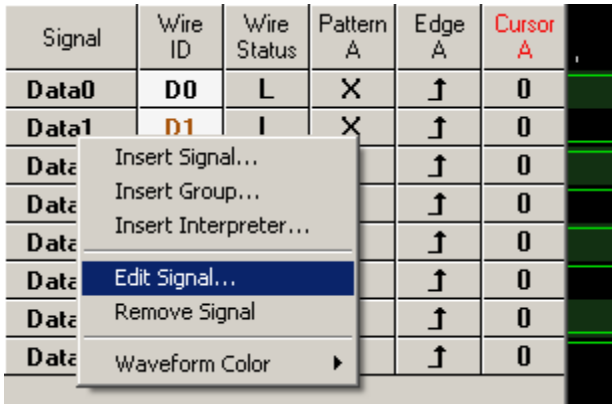
BAD



    - You can reorder stuff by just click and draging on the signal or group

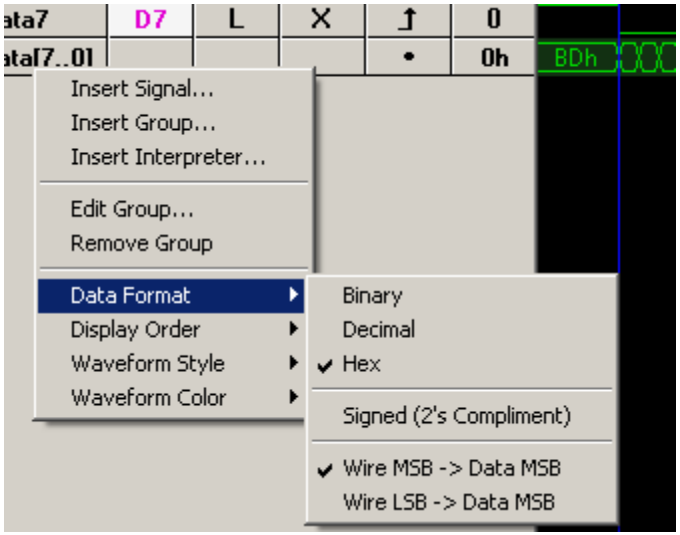
Reordering and Renaming

- A couple of things will make your setup more useful
  - Rename your signals/groups to something more useful
    - If one of you signals is an enable signal, rename it EN to make it easier to figure out what is going on, instead of trying to remember if Data3 or Data4 is the enable signal
  - Right click on the signal or group and click on ‘Edit Signal…’



Group Settings

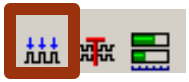
- If you right click on a group, you will see some group settings
  - Format
    - How the signals will be interpreted
    - Number base such as binary, hex, or decimal
    - 2’s compliment or not
    - Switch MSB or LSB order
  - Order
    - Which bit is displayed at the top of the group. This only changes the way it is displayed, not the way it is interpreted.
  - Style
    - Interpret the bits as a analog value
    - See the examples for … examples. (Slide 3)
  - Color
    - You can change the color! Woo…



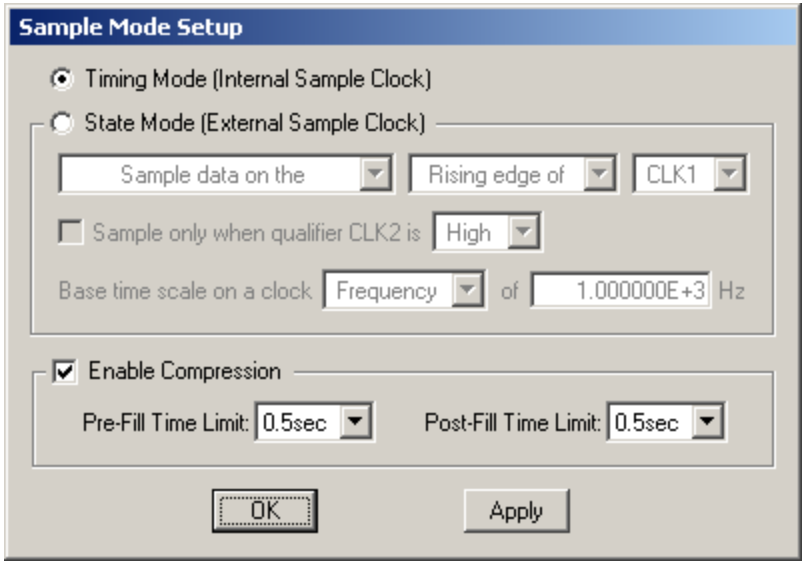
Software Setup

- Every time you intend to make a set of measurements you need to set up:
  - sample mode
  - triggering condition
  - sample rate
  - logic threshold
- Optional
  - Measurement setup
  - Pre-trigger buffer

Sample Mode



- Select one of two different modes
  - Timing mode
    - Use the internal clock, sampling rate is chosen using the sample rate drop down menu on the main screen
  - State mode
    - Use an external line to say when to sample (this doesn't have to be a clock necessarily)



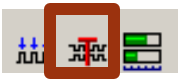
Sample Mode

- Timing Mode
  - Sampling is driven by an internal clock (100kHz -> 100K samples per second)
  - That sampling speed is chosen in the main window under using the Sample Rate drop down box (see sample rate slide)
  - Samples of every signal will be taken at this predefined rate
- State Mode
  - Sampling is driven by user defined settings and an external signal
  - The signal can only be from the CLK1 wire or the CLK2 wire.
- Which one you choose depends on what you are looking for

Sample Mode - Compression

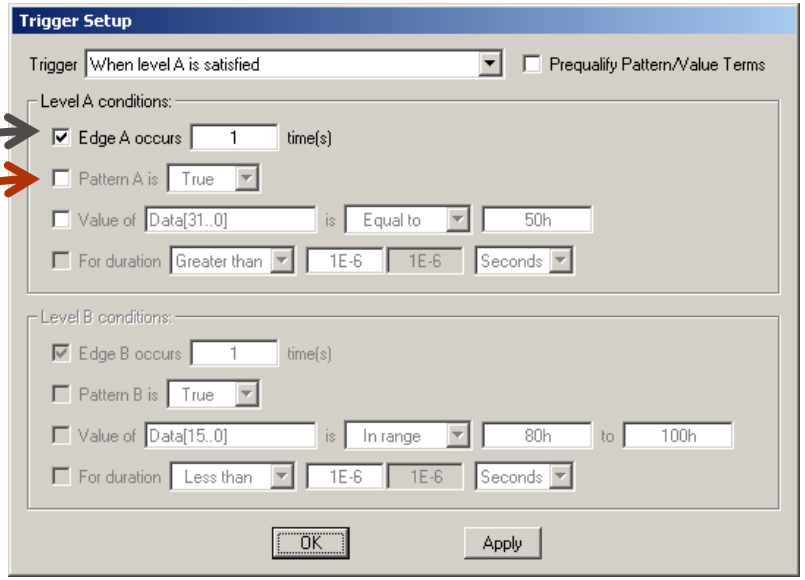
- The compression feature allows the buffer to hold more useful data
- When compression is enabled, the buffer will only hold data when transitions are detect, and fill in the static data in between later
- The suggestion is to always have this on

Triggering Setup



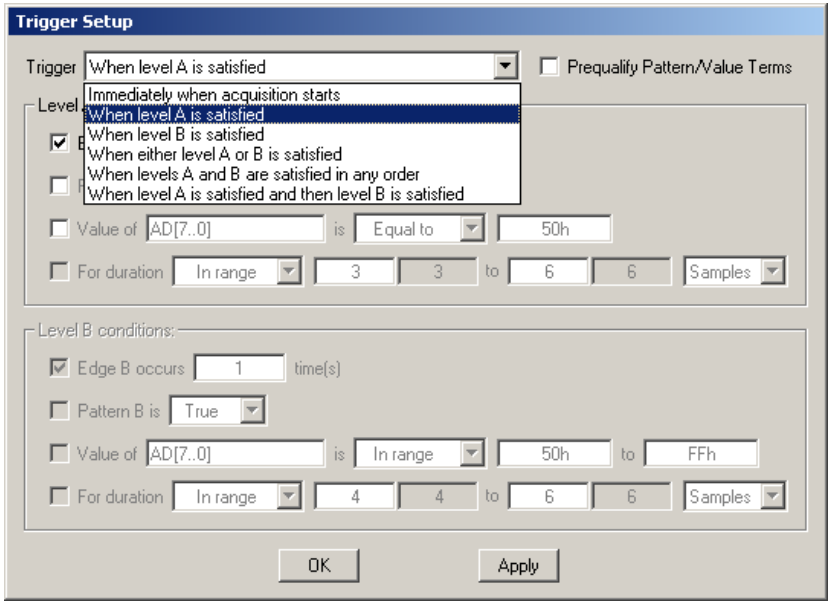
- The trigger determines when the buffer is sent to the computer
- Lots of choices
- The default is usually enough

Signal	Wire ID	Wire Status	Pattern A	Edge A	Cursor A
Clock1	CLK1	L	X		0
Clock2	CLK2	L	X		0
- Data[31..0]				•	0h
Data0	D0	L	X	↑	0
Data1	D1	L	X		0
Data2	D2	L	X		0
Data3	D3	L	X		0
Data4	D4	L	X		0
Data5	D5	L	X		0
Data6	D6	L	X		0
Data7	D7	L	X		0
Data8	D8	L	X		0



Trigger Setup

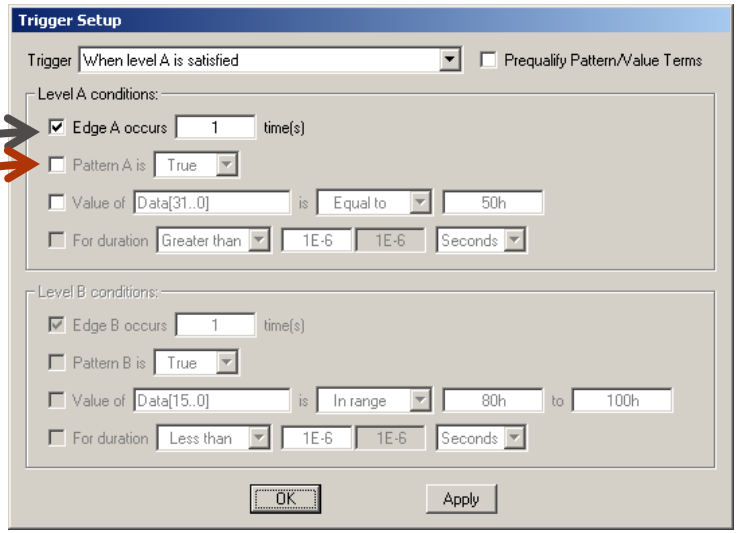
- You have two sets of conditions that you can specify (A and B)
  - Each condition can be a edge of a signal, a state of a bus or timing of a bus
- Additionally, you can say how the two conditions will effect the trigger



Trigger Setup

- Edge and patterns are defined in the waveform view by click on the related cell
  - Pattern: 0, 1, or don't care (X)
  - Edge: falling, rising, either, or none
  - Edges are logically OR'd (so a trigger would occur if the analyzer saw Data0 go low OR Data1 go high OR Data2 change at all)

Signal	Wire ID	Wire Status	Pattern A	Edge A
Clock1	CLK1	H	X	
Clock2	CLK2	H	X	
- Data[7..0]			•	•
Data0	D0	H	0	↓
Data1	D1	H	1	↑
Data2	D2	H	X	↕
Data3	D3	H	X	
Data4	D4	H	X	
Data5	D5	H	X	
Data6	D6	H	X	
Data7	D7	H	X	



Sampling Rate

- Since the buffer is a fixed size, higher sampling rate -> less actual time is measured
- The sampling rate is selected (when in timing mode) with the use of the drop down box

- Remember Nyquist
- Always think about what signal you expect, and set the sampling rate accordingly

Logic Threshold

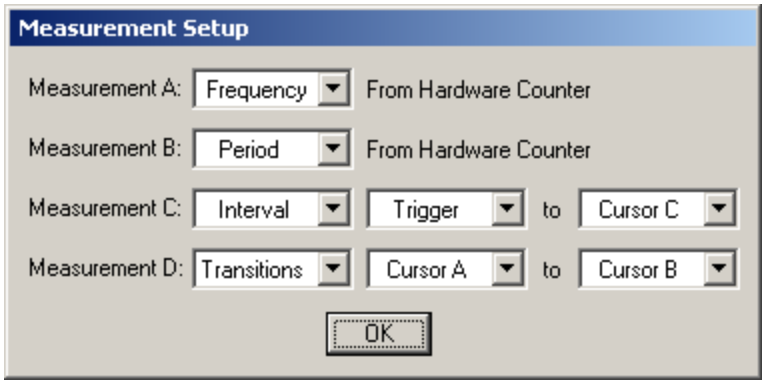
- The logic threshold determines where the analyzer will detect a high (binary 1) at and where it will detect a low (binary 0) at.
- If you set it to 1.4V, anything above 1.4V will be ‘1’, any thing below will be ‘0’.

- Again, always think about what you expect your signal to be, and set the threshold accordingly (e.g. this 1.4V threshold would be terrible for a 5V system.)

Measurement Setup

- You have four different measurements you can use, which are displayed at the bottom of the window.

- You can change what they are and how they are calculated by using the measurement setup screen.
- The in-software help is useful if you want to know more.



Pre-Trigger Buffer

- What portion of the buffer should be viewable before and after the trigger event

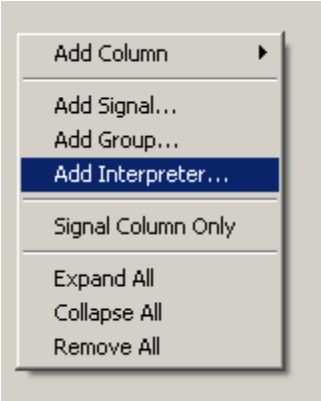
- If the pre-trigger buffer is 50%, the buffer displayed will have the trigger event in the middle of the buffer data
- If it is at 7%, 7% of buffer data will be before the trigger
- Note: compression will significantly change how this works though, since buffer usage is calculated after compression

Interpreters

Interpreters

- Interpreters are
  - Awesome things which will save you lots of time…
  - The software is able to look at the data, and interpret it using certain communication protocols

- To use one, right click on the area under your signals

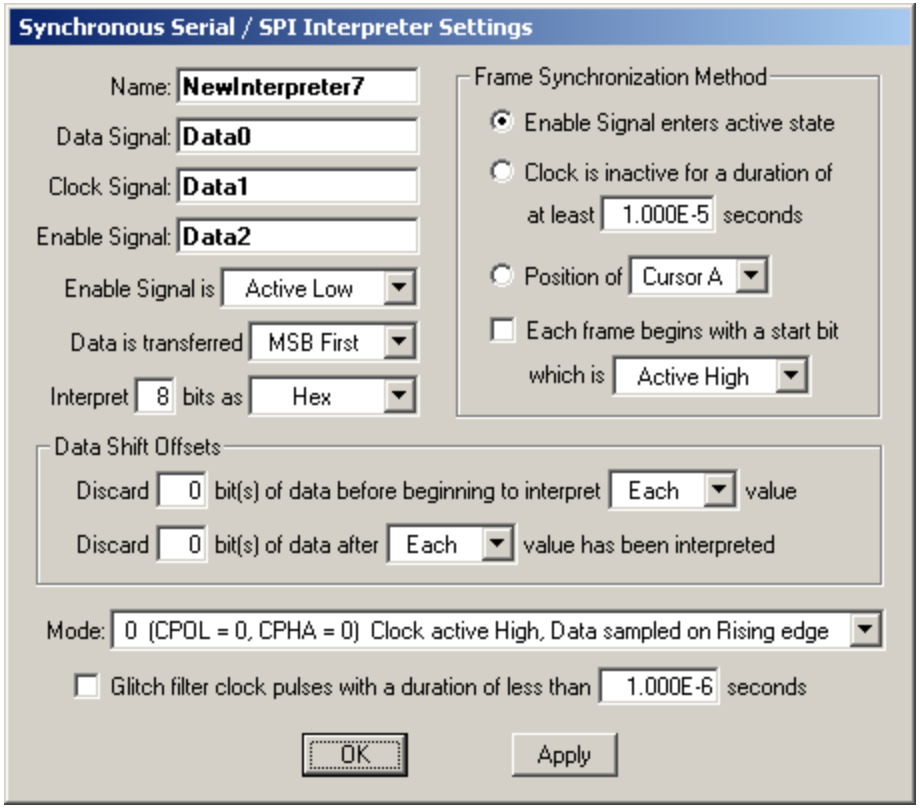


Interpreters

- You have a couple of options
  - I2C, SPI, RS232 or CAN interpreters
- Similar to groups, you can
  - Have multiple interpreters, but not two of the same exact one (but you can have two of the same kind, e.g. two RS232 interpreters you just need to make another one, so RS232_A and RS232_B)
  - Create new interpreters

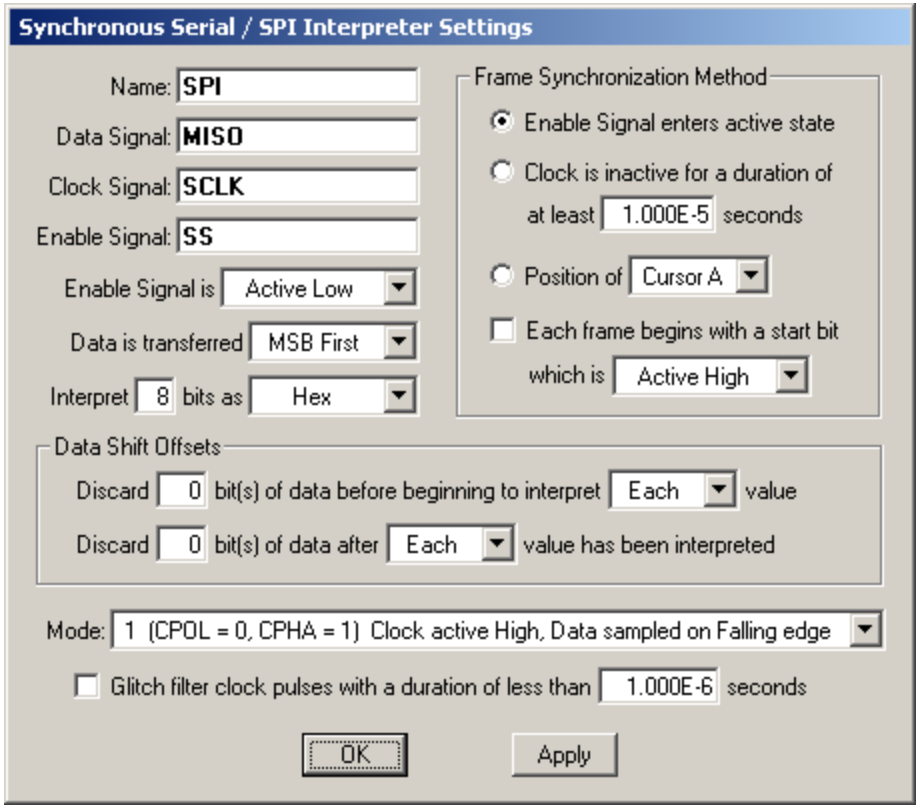
Interpreter Example

- Lets make a new interpreter
  - Right click -> Add Interpreter -> Create… -> SPI
  - Rename it to useful: SPI or MSP430 SPI
  - You can change the signals assigned to the interpreter signals, just click on the names
  - For any interpreter, always check all these setting to make sure they match up with what your protocol should be doing
  - Synchronization is very important, or else the interpreter will get confused on when to start interpreting, this is a usual area of trouble


Synchronous Serial / SPI Interpreter Settings
Name: NewInterpreter7
Data Signal: Data0
Clock Signal: Data1
Enable Signal: Data2
Enable Signal is Active Low
Data is transferred MSB First
Interpret 8 bits as Hex
Frame Synchronization Method
Enable Signal enters active state
Clock is inactive for a duration of
at least 1.000E-5 seconds
Position of Cursor A
Each frame begins with a start bit
which is Active High
Data Shift Offsets
Discard 0 bit(s) of data before beginning to interpret Each value
Discard 0 bit(s) of data after Each value has been interpreted
Mode: 0 (CPOL = 0, CPHA = 0) Clock active High, Data sampled on Rising edge
Glitch filter clock pulses with a duration of less than 1.000E-6 seconds
OK
Apply

Interpreter Example

- Rename the signals (click on the signal name, then in the new window, right click on the signal)
- The SPI bus will be operating in mode 1, so change the mode at the bottom of the window
- Our frame synchronization method should be fine, so leave it as is


Synchronous Serial / SPI Interpreter Settings
Name: SPI
Data Signal: MISO
Clock Signal: SCLK
Enable Signal: SS
Enable Signal is Active Low
Data is transferred MSB First
Interpret 8 bits as Hex
Frame Synchronization Method
Enable Signal enters active state
Clock is inactive for a duration of at least 1.000E-5 seconds
Position of Cursor A
Each frame begins with a start bit which is Active High
Data Shift Offsets
Discard 0 bit(s) of data before beginning to interpret Each value
Discard 0 bit(s) of data after Each value has been interpreted
Mode: 1 (CPOL = 0, CPHA = 1) Clock active High, Data sampled on Falling edge
Glitch filter clock pulses with a duration of less than 1.000E-6 seconds
OK
Apply

Interpreter Examples

- Click OK to go back to the interpreter selection window
  - Note: you can delete an interpreter by right click on a selected interpreter
- Click on the newly created SPI interpreter and say OK
- You can now see the interpreter and its associated signals.

Cursors

- Useful for the measurements (time between two events) or looking at the data at certain times (Cursor A column shows the data where cursor A is located)
- You can click and drag any of the six cursors
- Right click in the waveform or state list window -> **place cursor** or **scroll to cursor**
- Note: hit ‘A’ to go to the A cursor (or ‘T’ to go to the trigger event)

Getting Data Out

- Couple of options
  - Image: Ctrl+I (File->Print to Clipboard)
  - CSV file: Ctrl+D (File->Export Data…)
  - Print: Ctrl+P (File->Print…)

Troubleshooting

- **The analyzer is not triggering**
  - Is the voltage level set correctly for your signal?
  - Do you actually have a trigger setup on the correct signal(s)?
  - Is your sample rate high enough?
- **The analyzer is triggering, but nothing shows up**
  - Is you sample rate high enough or is it too high?
  - The sample rate could be to high, and you are just not seeing enough time to see the result after the trigger

Troubleshooting

- **The interpreter shows nothing, or it is wrong**
  - Are all the signals appropriately assigned to their interpreter counter part? (Right click interpreter, edit interpreter)
  - Is your frame synchronization method appropriate for the setup? For example, if your enable never changes (always enabled), you should not use the “Enable signal enters active state” for the synchronization method.
  - Do you have the correct settings, such as active low/high enable signal, MSB or LSB transfer order, baud rate, number of bits, etc. (This changes drastically for the different protocols.)

Other Resources

- ECEN 5613 LogicPort Introduction
  - http://ecee.colorado.edu/~mcclurel/LogicPortAnalyzerNotes_4-30-2011.pdf

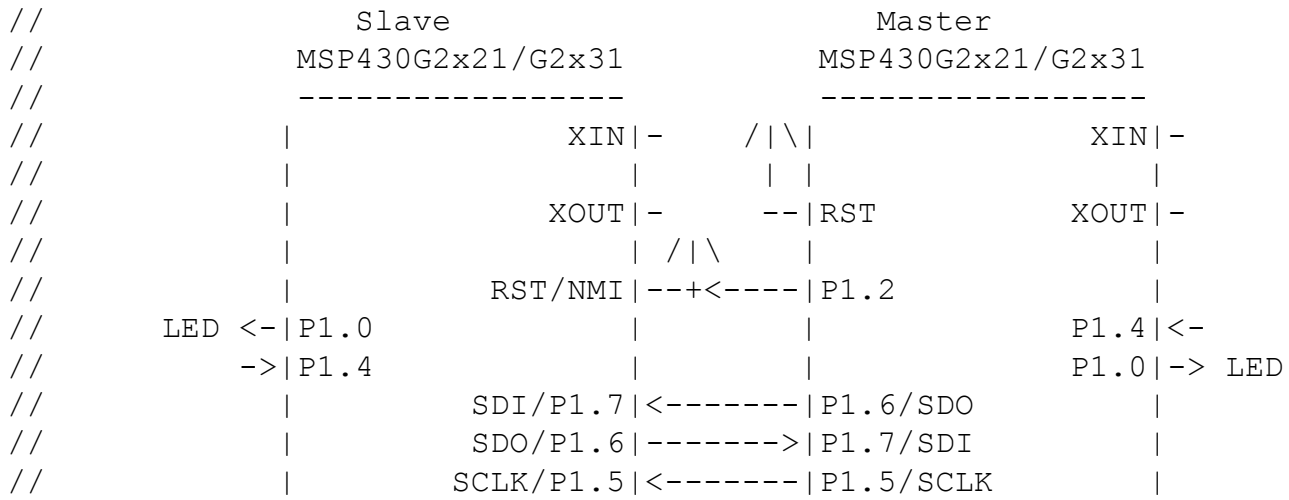
SPI Example with MSP430

Example

- Using
  - LogicPort analyzer and software
  - MSP430 LaunchPad (which has the MSP430G2231)
  - CodeComposer v4

Example

- From the code example (we are going to just program the master):

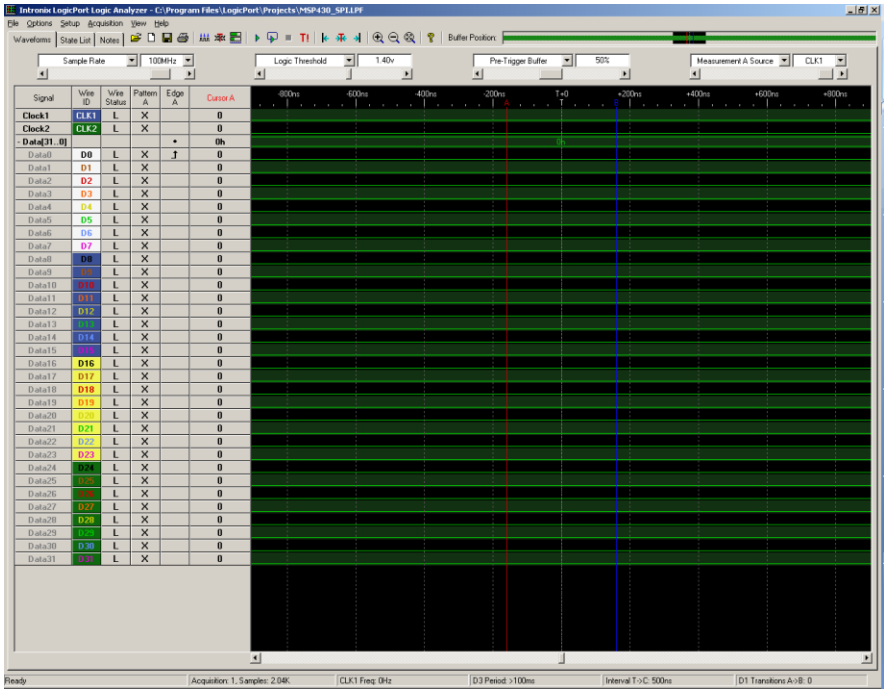


Example

- Step 1, think about what you expect:
  - The code I am working with will generate a SPI communication packet at something around 100KHz
  - The signal will be a 0V-3.3V signal
  - The SPI will be in mode 0, and I have no chip select/slave select signal
- Knowing these, lets set up out environment

Example

- Since I will want to use this setup again, I'm going to make a new file (File->New)
- I'm going to call mine MSP430_SPI
- We don't need all of these signals, so remove them all.


Intronix LogicPort Logic Analyzer - C:\Program Files\LogicPort\Projects\MSP430_SPI.LPF
File Options Setup Acquisition View Help
Waveforms State List Notes
Buffer Position:
Sample Rate 100MHz
Logic Threshold 1.40v
Pre-Trigger Buffer 50%
Measurement A Source CLK1
Signal Wire ID Wire Status Pattern A Edge A Cursor A
Clock1 CLK1 L X 0
Clock2 CLK2 L X 0
- Data[31..0] 0h
Data0 D0 L X 0
Data1 D1 L X 0
Data2 D2 L X 0
Data3 D3 L X 0
Data4 D4 L X 0
Data5 D5 L X 0
Data6 D6 L X 0
Data7 D7 L X 0
Data8 D8 L X 0
Data9 D9 L X 0
Data10 D10 L X 0
Data11 D11 L X 0
Data12 D12 L X 0
Data13 D13 L X 0
Data14 D14 L X 0
Data15 D15 L X 0
Data16 D16 L X 0
Data17 D17 L X 0
Data18 D18 L X 0
Data19 D19 L X 0
Data20 D20 L X 0
Data21 D21 L X 0
Data22 D22 L X 0
Data23 D23 L X 0
Data24 D24 L X 0
Data25 D25 L X 0
Data26 D26 L X 0
Data27 D27 L X 0
Data28 D28 L X 0
Data29 D29 L X 0
Data30 D30 L X 0
Data31 D31 L X 0
-800ns -600ns -400ns -200ns T+0 +200ns +400ns +600ns +800ns
Ready
Acquisition: 1, Samples: 2.04K
CLK1 Freq: 0Hz
D3 Period: >100ns
Interval T->C: 500ns
D1 Transitions A->B: 0

Example

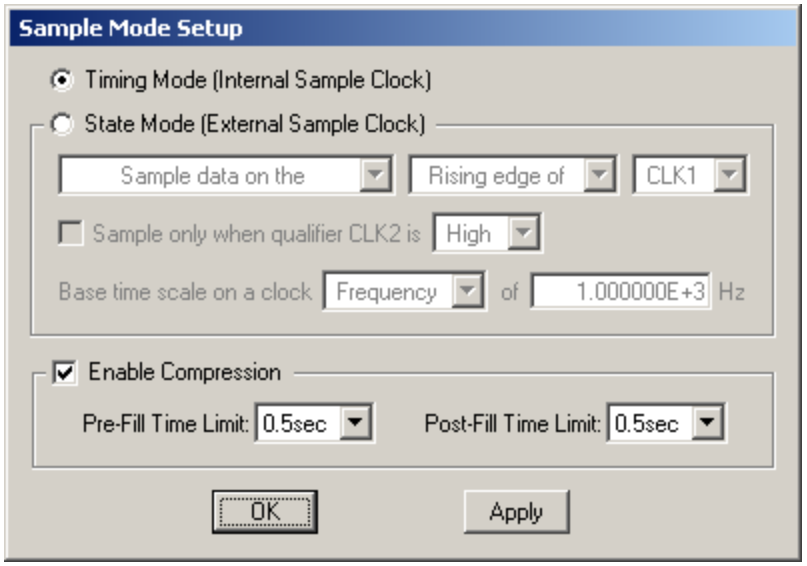
- The code has SCLK on pin P1.5, and SDO (data out) on pin P1.6.
- So, I need 3 signals (the default names will be used until I rename them):
  - Data0 -> SDO
  - Data1 -> SCLK
  - GND -> GND
- Since the interpreter also expects an enable signal, I will also connect Data2 -> GND just so it isn’t doing weird things (but it won’t be used)

Example

- So in my window I want to setup a couple of things:
  - Setup the sample mode
  - Setup the trigger mode
  - Sampling rate
  - Logic threshold
  - Add an interpreter

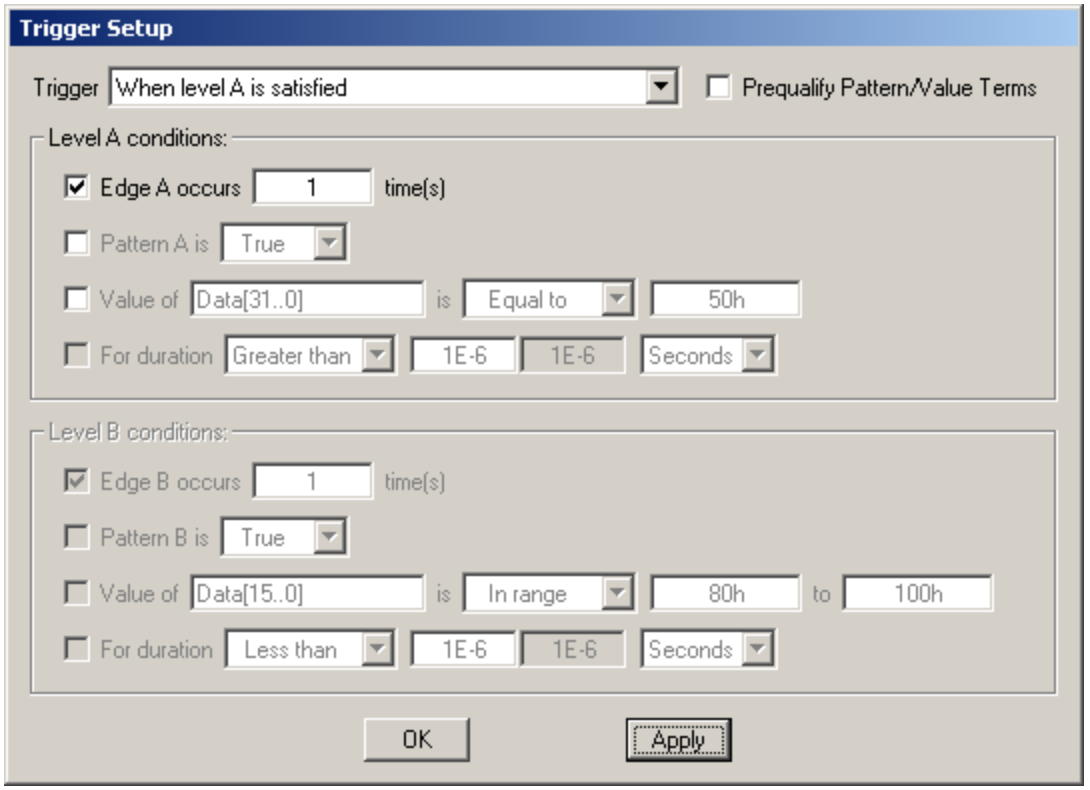
Sample Mode

- Timing Mode will be fine for this application



Trigger Mode

- I will be letting the edge of a clock be the trigger mechanism, so I can set up the following conditions:



Sampling Rate and Logic Threshold

- I know my SPI clock will be going at about 100KHz, so I need at least a sampling rate of 200KHz
  - I set it to 500KHz just to be safe
- I am on a 3.3V system, so I will set the logic threshold to about ½ of that, 1.4V

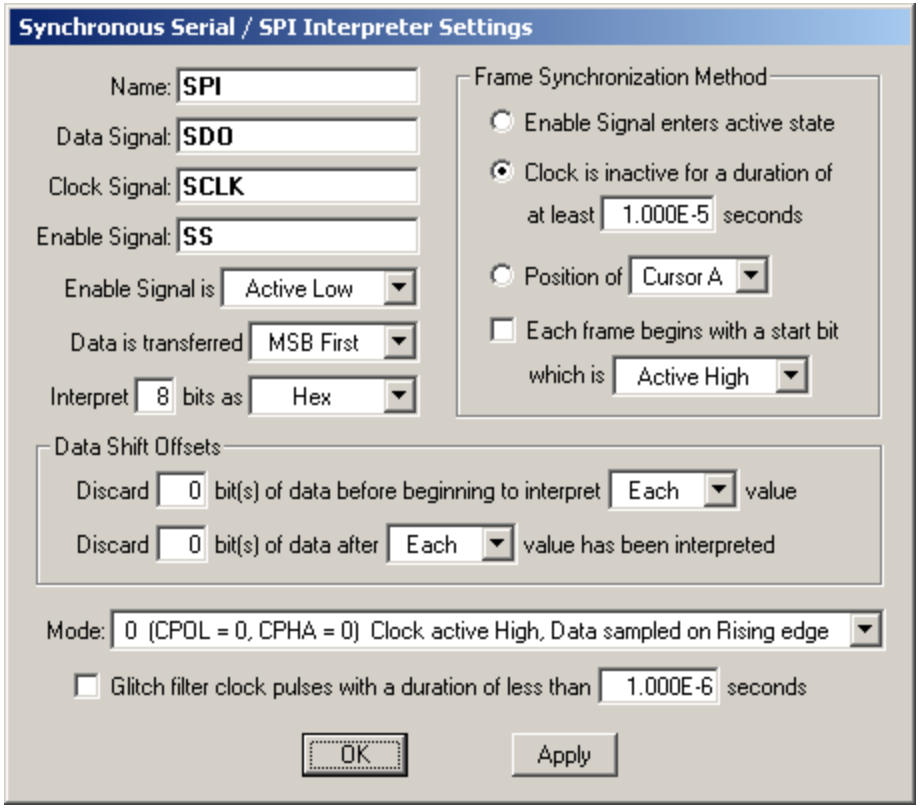
- The standard 50% Pre-Trigger buffer will be fine for now
- I'm not going to mess with any of the measurements now

Add an Interpreter

- I want SPI, so I will make a new interpreter for SPI which has the following settings:

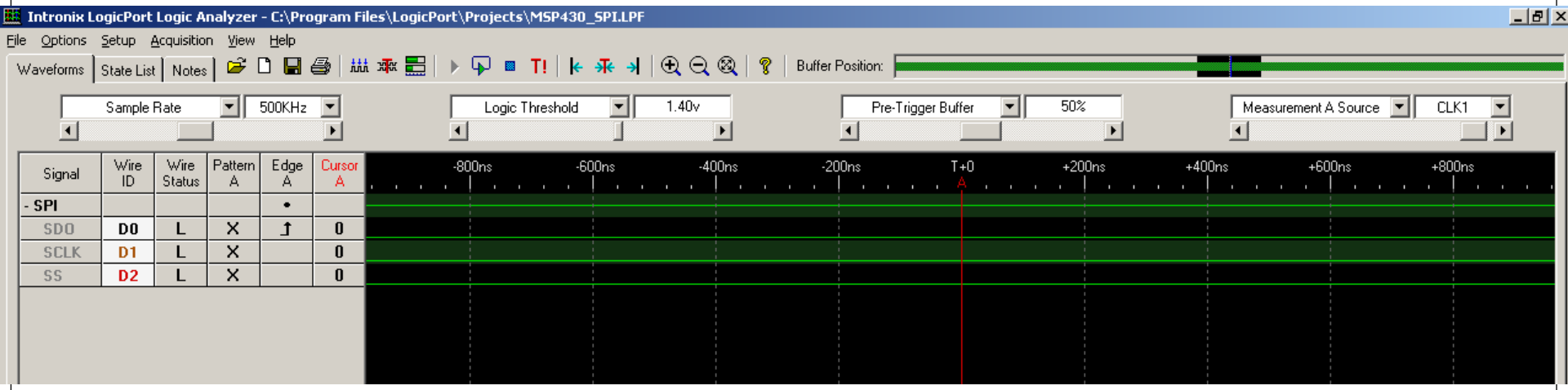
Notice the Frame Synchronization Method and the Mode. Since my enable signal will always be low, I can't use 'enable signal enters active state'.

I also renamed the signals to make it easier to read.


Synchronous Serial / SPI Interpreter Settings
Name: SPI
Data Signal: SDO
Clock Signal: SCLK
Enable Signal: SS
Enable Signal is Active Low
Data is transferred MSB First
Interpret 8 bits as Hex
Frame Synchronization Method
Enable Signal enters active state
Clock is inactive for a duration of at least 1.000E-5 seconds
Position of Cursor A
Each frame begins with a start bit which is Active High
Data Shift Offsets
Discard 0 bit(s) of data before beginning to interpret Each value
Discard 0 bit(s) of data after Each value has been interpreted
Mode: 0 (CPOL = 0, CPHA = 0) Clock active High, Data sampled on Rising edge
Glitch filter clock pulses with a duration of less than 1.000E-6 seconds
OK
Apply

Setup

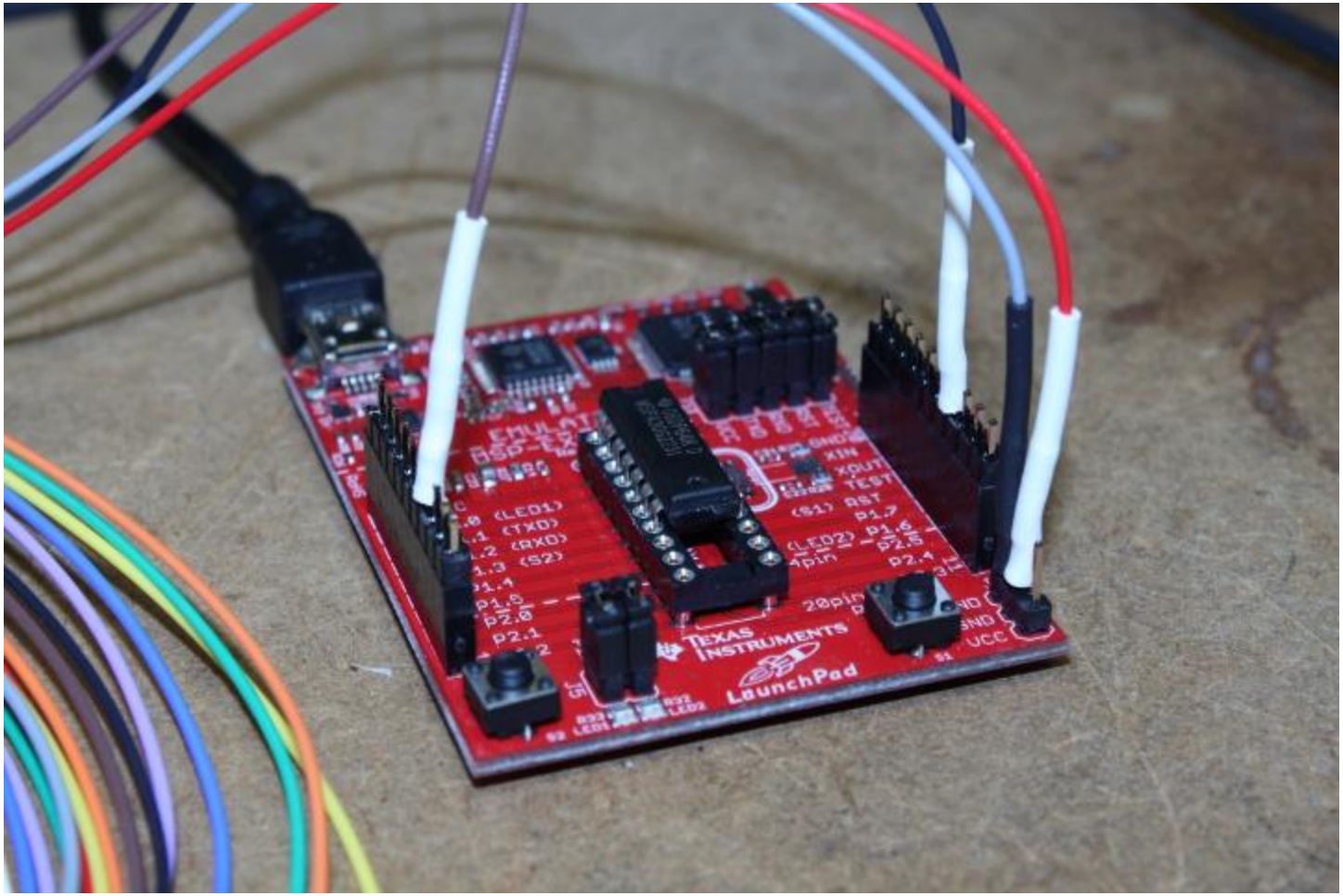
- Should look like this:


Intronix LogicPort Logic Analyzer - C:\Program Files\LogicPort\Projects\MSP430_SPI.LPF
File Options Setup Acquisition View Help
Waveforms State List Notes
Buffer Position:
Sample Rate 500KHz
Logic Threshold 1.40v
Pre-Trigger Buffer 50%
Measurement A Source CLK1
Signal Wire ID Wire Status Pattern A Edge A Cursor A
- SPI
SDO D0 L X 0
SCLK D1 L X 0
SS D2 L X 0
-800ns -600ns -400ns -200ns T+0 +200ns +400ns +600ns +800ns
A

- Notice the Edge A has already been set as the rising edge of SDO (Data0)

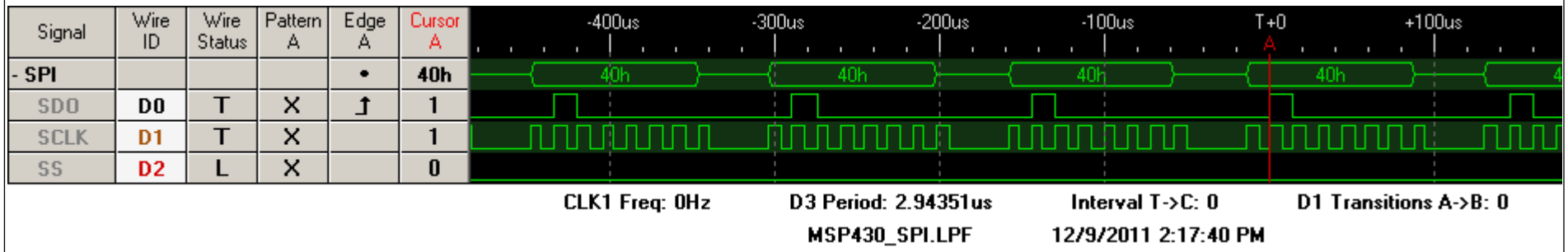
Hardware Setup

- Attach wires to appropriate pins…



Single Acquisition

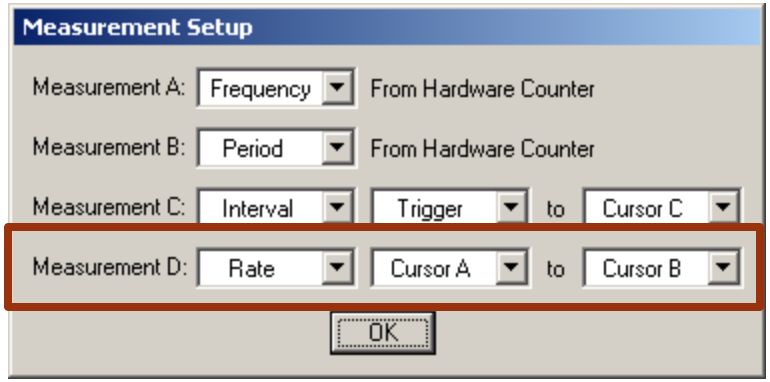
- Click on the little green arrow and hopefully you get something like this:



- Happily, this is what I expected

Use the cursors

- If I place the cursor A at a rising edge of a clock and cursor B at the next rising edge I can figure out what the SPI clock frequency is:


Measurement Setup
Measurement A: Frequency From Hardware Counter
Measurement B: Period From Hardware Counter
Measurement C: Interval Trigger to Cursor C
Measurement D: Rate Cursor A to Cursor B
OK